Use of UML

Jim Kowalkowski

Personal Case Tool Statement

- Experience shows that this environment does not lend itself to a heavy-weight development process and therefore should be avoided as a project standard
- UML (Unified Modeling Language) is useful outside a heavy development process

Diagram Types

- Class diagrams
  - Static in nature, not concerned with objects in the running program
- Sequence diagrams
  - Relates method calls of objects in time
  - Shows the time at which method calls are made
- Object diagrams
  - Relationships of objects in a running system
- State diagrams
- Many others

Interesting Class Diagram Notation

- Classes, methods, and data members
- Inheritance
- Relationships including expression of ownership
- Qualities such as constant
- Parameterized classes
- Using relationship
- Notes
- Hierarchical structuring of classes (packages)

Where it is Useful

- Capturing *key design concepts*
- Showing library or subsystem design
- Discussions and walkthroughs before coding starts
- Reviews of already designed projects
- Roadmap for implementing the subsystem

Where it Helps

- Preventing an implementation that does not satisfy the requirements
- Preventing an explosion of classes and utilities from on-the-fly design
- The relationship of classes to other parts of the system becomes visible

Where it has not Been Useful

- Capturing *all* implementation details
- Generating code from the diagrams
  - The C++ code from the tool will likely not be your style
  - Mapping all the notation into C++ syntax is complex
- Attempting to be very complete in the use of notation
- Reverse engineering

Reverse Engineering

- Relationships are interpreted incorrectly
- Inexperience with C++ hurts
  - Arrays as pointers (int*)
  - Vectors of pointers to objects
- Division or grouping of classes is not correct
- Rat’s net

Difficulties

- Keeping the diagrams in sync with the code after the initial implementation is complete
- Expressing some generic programming concepts
  - type lookup within a class

Summary

- Great for capturing ***key*** design elements
- Should be used at the early stages of a package or library
- Not good for capturing every implementation detail
- Not good for reverse engineering
- Not good for code generation
- Not good for driving the entire development cycle

Current Recommendations

- Visio2000
  - *http://www.microsoft.com/office/visio/*
- *The Unified Modeling Language User Guide*, G.Booch, et al., Addison Wesley, 1999.